**What is Stack Overflow? Not the website but the concept.**

Till recently, I always had a hard time understanding what stack overflow meant. Not the website but the concept.

Through a fortuitous conversation with an enlightened colleague, I was finally able to understand what this concept means.

To understand **stack overflow** we first have to understand what the **call stack** or **stack**means.

So, most programming languages support the concept of functions. In fact I'd be as bold to say that almost all mainstream languages support the concept of functions. And languages such as Clojure, Haskell, LISP only support the concept of functions, these are referred to as functional languages.

A function is a named piece of code. Like when you define a function, you assign it a name and the statements or code comprising the function is stored in the memory that is allocated to the function.

So, when you create a variable you are setting aside some space in memory to hold a value, say a number or a string. And when you define a function you're setting aside some memory which holds the statements comprising the function. And this memory location can be accessed through the function name, just like variable names allow us to access the memory locations containing their values.

So the **call stack**or the **stack**is a segment of memory or a part of the main memory or RAM (Random access memory, the kind of memory which is used for processing when you switch on your system and run some applications) which is set aside for storing the arguments to and return values from a function call.

So if you have functions you'll have a call stack.

Whenever a function call is made, the arguments for that function and the return value location are passed to the call stack.

And as multiple functions are called, the arguments and return addresses are stored in the stack and the stack grows.

And as more and more function calls are made, more and more of the stack is taken up and at a certain point, the stack goes out of space. That is, there is no memory available on the call stack to store another function call. And this condition is called **stack overflow**.

Now, **stack overflow** error or condition is most common with the usage of recursion. For an improperly defined recursive function with no base case, the function would be called infinitely and of course it can't be called infinitely because there's not enough stack memory => stack overflow.

The wikipedia link is really good:

<http://en.wikipedia.org/wiki/Stack_overflow>

I liked this link on debugging a stack overflow error because it presents steps for analysis of the error, may not be the same situation on different platforms and for different languages, but the general principles would remain similar:

<https://msdn.microsoft.com/en-us/library/windows/hardware/ff540620(v=vs.85).aspx>

*— MAR 07, 2015 —*